Electric Vehicle Charging Demand Forecasting Model Technical Documentation

Postcode-Level Daily kWh Forecasting Pipeline

submitted to

Deakin University
25 January 2026

by
William Connor Allchin

1. EXECUTIVE SUMMARY

Electric Vehicle Charging Demand Forecasting Model

This document describes a machine learning pipeline designed to predict daily electric vehicle (EV) charging demand at the postcode level across Australia. The model forecasts energy consumption in kilowatt-hours (kWh) based on geographic location, temporal factors (e.g. day of week), and weather conditions.

Key Capabilities:

- Predicts daily EV charging demand for any Australian postcode
- Incorporates real-time weather forecasts (up to 16 days ahead)
- Accounts for seasonal variations, public holidays, and weekend patterns
- Achieves a Root Mean Squared Error (RMSE) of 7.65 kWh on test data

Data Sources:

- CSIRO Draft 2026 Forecasting Assumptions (vehicle efficiency and driving patterns)
- Australian Automobile Association (vehicle registration distribution)
- NOAA GHCN Daily (historical weather data)
- Open-Meteo API (weather forecasts)
- Holiday API (Australian public holidays)

Model Performance:

- Training Period: January 1, 2023 – December 21, 2025
- Test Period: December 22, 2025 – January 20, 2026
- Training Samples: 2,750,838 rows
- Test Samples: 75,990 rows
- RMSE: 7.6505 kWh
- MSE: 58.5295

2. DATA SOURCES

I used the below data sources because they are free and accessible. I have no doubts that paid versions may have better coverage, and would result in a more effective model.

CSIRO Vehicle Efficiency Data

Source: [CSIRO Draft 2026 Forecasting Assumptions Update (December 2025)]

Data Included:

- Estimated driving distance by vehicle type (km/year)
- Energy efficiency by vehicle type (kWh/km)

Vehicle Types:

Vehicle Type (Code)	Translation
RES_BIKE	Bike Residential
RES_SMALL	Small Residential
RES_MEDIUM	Medium Residential
RES_LARGE	Large Residential
COM_LCV_SMALL	Small Light Commercial
COM_LCV_MEDIUM	Medium Light Commercial
COM_LCV_LARGE	Large Light Commercial
COM_RGD_TRUCK	Rigid Truck
COM_ART_TRUCK	Articulated Truck
COM_BUS	Bus

CSIRO Forecast Vehicle Numbers By State and Type
Contains excel sheets full of data, including total number of BEVs, PHEVs and hybrids by state (and vehicle type).

Source: [Draft 2026 Forecasting Assumptions Update Detailed EV Workbook]

Import instructions for extracting this data for working the notebook (if not using the .csv files provided in the github repo, or if needing to update with data from newer workbooks by CSIRO):

1. Download the workbook
2. Save the “Scenario”, “Region”, “Vehicle Category”, and “2025-26” columns from each of the “BEV_Numbers”, “PHEV_Numbers”, and “Hybrid_Numbers” tables to .csv files
3. Name them “BEV_numbers_25-26.csv”, “PHEV_numbers_25-26.csv”, and “hybrid_numbers_25-26.csv”, respectively (or update pd.read_csv() code)
4. Run the notebook to load them in

AAA Vehicle Registration Data

Source: Australian Automobile Association

Data Included:

- Geographic distribution of vehicle registrations (by postcode)

Weather Data

Historical Data Source: NOAA Global Historical Climatology Network (GHCN) Daily

Forecast Data Source: Open-Meteo API

Weather Features:

- Daily mean temperature (average of TMAX and TMIN)
- Forecast range: Up to 16 days ahead
- Temperature units: Celsius

Public Holidays

Source: Holidays Python library

Coverage: All Australian public holidays (federal and state-level)

3. TECHNICAL DEPENDENCIES

Required Python Libraries:

pandas >= 1.3.0
numpy >= 1.21.0
lightgbm >= 3.3.0
scikit-learn >= 1.0.0
holidays >= 0.13
scipy >= 1.7.0
tqdm >= 4.62.0
requests >= 2.26.0
joblib >= 1.1.0

I strongly recommend installing these in a new environment using a system package manager like conda. It will work on the latest versions of the above packages as of 25/01/2026.

API Requirements

- Open-Meteo API: No authentication required (free tier)
- Holiday API: API key required for production use
- NOAA GHCN: Public access, no authentication

4. MODEL ARCHITECTURE

Model: LightGBM Regressor (Light Gradient Boosting Machine)

Rationale:

- Native support for categorical features (Postcode, State)
- Fast training and prediction
- Handles large datasets efficiently
- Robust to overfitting with proper tuning

Feature Importance Considerations:

- *baseline_daily_kwh*: Primary driver of demand magnitude
- *temperature*: Captures weather-related efficiency variations
- *is_holiday* and *is_weekend*: Capture behavioral patterns
- Temporal features (day, month, etc.): Capture seasonal trends

Hyperparameter Tuning

Method: Random Search with Time Series Cross-Validation

Search Space:

- learning_rate: Uniform distribution (0.01, 0.19)
- n_estimators: Integer range (50, 1000)
- max_depth: Integer range (3, 15)
- num_leaves: Integer range (20, 150)

Cross-Validation: TimeSeriesSplit with 5 splits (maintains temporal order)

Optimisation Metric: Negative Mean Squared Error

Best Parameters Found:

- learning_rate: 0.05783863620681919
- n_estimators: 436
- max_depth: 4
- num_leaves: 132

Tuning Duration: Approximately 1 hour on an AMD Ryzen 7 8845HS

Train/Test Split: Time-based split (last 30 days as test set) to simulate real-world forecasting

5. LIMITATIONS AND ASSUMPTIONS

Limitations:

- Forecast Horizon: Weather forecasts limited to 16 days ahead
- Synthetic Training Data: Model trained on synthetic demand, not actual demand (as it does not exist publicly)

Assumptions:

- EV charging demand correlates with daily driving distance
- Holiday and weekend patterns are consistent across regions
- Vehicle efficiency estimates from CSIRO are accurate

6. SAVED MODEL ASSETS

The trained model and supporting files are saved in the '/demand_forecasting_api/' folder:

- ev_demand_model.pkl (trained model)
- postcode_baseline.csv (baseline daily kWh demand per postcode)
- postcode_coords.csv (coordinates for each postcode)
- feature_columns.txt (comma-separated list of feature names (ensures correct order))

7. USING THE API

In order to run the API locally on your machine, you can follow these steps:

1. In your command prompt/terminal, navigate to '/demand_forecasting_api/'.

2. Install the libraries inside *requirements.txt*.
   e.g.: pip install -r requirements.txt

3. While still inside '/demand_forecasting_api', run the following:
   uvicorn api:app --reload

4. Open your web browser and go to http://127.0.0.1:8000/docs. FastAPI automatically creates an interactive documentation page where you can test the API by entering a postcode and date and clicking *Execute*.

Note: You may need to adjust the above commands depending on the terminal you use.